ONTEREST
Software Test Report

1. Introduction

1.1 Purpose and Scope

The Software Test Report is a document that contains a summary of all testing activities and the final test results of the project. A test report is an assessment of how well the tests were performed. Stakeholders can assess the quality of the tested product and use the test report to decide whether or not to release the software.This document is STR of website Onterest software development project.

Functional testing for the following modules are in scope of testing.

- Sign up
- Login
- Add, list, delete and update user(admin,sub-admin,member)
- Add, list and update club/sub-club
- Add, delete, list users to a club/sub-club

1.2 Descriptions and Abbreviations

STR	Software Test Report
UC	Use Case
ONT	Onterest
PM	PostMan
WB	MySQL Workbench
TCD	Test Case Definitions
SRS	System-Wide Requirements Specification
CCM	Configuration and Change Management
RM	Risk Management
SDD	Software Design Description
UCD	Use Case Definitions
AN	Architecture Notebook

2. Descriptions of Test Activities

2.1 Document Security

This document contains classified information about development of project Onterest. Therefore it shouldn’t be shared with third parties.

2.2 Data Recording, Reduction and Analysis

While testing our project, we used PostMan application mainly. Most of our tests are manual and semi-automatic. For example after creating a user manually, we check everything about that user from database. The database creation is otomatic but we check the values ourselves so these tests are semi-automatic. We used PostMan to verify whether our get, post, delete and update functions are working. We saved our test results for this document. We analyzed and notified developers(ourselves) for failed test. This part is called bug fixing.

3. Test Preparation

3.1 Hardware Preparation

We used PostMan for testing and MySQL Workbench for set up our database. We used Maven libraries as well. Our projects runs on JDK 8, so this must be downloaded and configured as well. After installation of all of these and setting up the project correctly, WB is used to set up the database for onterest. When preperation are finally done, PM can be used to test API and database testing. We also test our interface to see if we can send requests to database correctly and get correct results. Since our website works on a localhost, we have no problems with servers. Steps to run database tests:

1-Open PM

2-Skip sign-up/login page

3-Press “+” to create requests

4-Choose a function to test

5-Enter request URL with localhost adress and which part of the onterest to test

Example: localhost:8080/api/members

3.2 Software Preparation

Testing stage is important for software robustness.In order for the product to be used for a long time, the testing phase must be done correctly. Problems such as poorly prepared test scenarios, insufficient test numbers, not adjusting the test environment are the main testing problems.

In this project we keep our data in our local data storage units. When we go online, our users can reach our product via internet. They do not need any external or internal stroge for using Onterest.

For backend side all communications are provided by HTTP requets. SO, we use POSTMAN software to check our queries. POSTMAN has high functionality and pretty good GUI, all CRUD operations can be tested without coding.

3.3 Other Pre-test Preparations

None.

4. Test Results

Use Case Number	Use Case Name	Test Case Description	Retest?	Number of Activity Paths Tested	Number of Non-working Activity Paths	The time spent (minutes)
1	Sign Up	Verify registration to system is successful	YES	6	1	20
2	Log in	Verify log in is successfful	YES	7	1	10
3	Forgot Password	Verify correct actor password is sent to correct mail	YES	7	3	10
4	New club categories	Verify new club/sub-clubs can be opened	NO	5	3	15
5	Member management	Verify actor can remove an actor from a club/sub-club	NO	3	2	5
6	Account management	Verify actor can manage their account settings and info	YES	3	0	20
7	Social club management	Verify social clubs are managable by admins	NO	5	4	5
8	Moderation	Verify moderation is given to only applicable accounts and if user bans work	NO	5	0	10
9	System Bug Reports	Verify actor is able to send bug reports	YES	6	1	10
10	Ontry	Verify actor can post threads	YES	5	0	10
11	Member Report	Verify actor is able to report inappropriate member behaviour	NO	6	0	5
12	Search Club/Sub-club	Verify actor can search for a club-sub/club	NO	4	4	5
13	Search members in the same sub-club	Verify a user can search for a member in the same sub-club	NO	3	3	5
14	Comment	Verify actor is able to comment on threads	NO	4	4	5

15	Establish Questionnaire	Verify a questionnaire is established	NO	5	1	10
16	Create event	Verify sub-club admin can create event	NO	4	1	5
17	Send message	Verify actor can send message to other actor	NO	2	1	2
18	Reviews and ratings	Verifuy reviews and ratings for clubs/sub-clubs are available	NO	2	0	1
	TOTAL	***18***		***82***	***29***	***153***

4.1 Criteria for Evaluating Results

Once the testing phase is complete, don't expect to be successful. Some tests pass, some fail. The criteria you consider while testing are also very important. The success factor is determined according to these criteria.

General Criterias

- We are now in the age of internet and speed. Therefore, it is important that the software works fast. Every transaction should take place within milliseconds.
- System must be sustainable. Even if an error is encountered, the system should continue to work without crashing.
- There should be warnings for every unexpected situation

Special Criterias

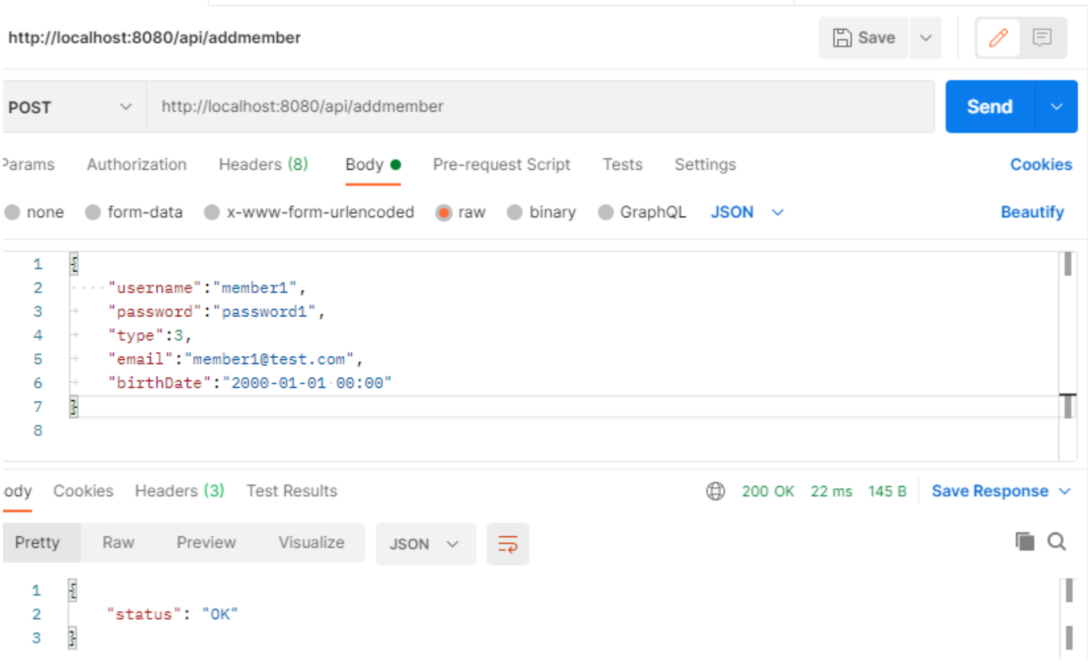
- You have to indicate your date of birth to become a member. (Sign In)
- -According to the site rules, you should not talk about religion, politics, racism and sexuality.(membership)

5. References

- ONT SRS Document
- ONT TCD Document
- ONT UCD Document
- ONT CCM Document
- ONT AN Document
- ONT RM Document
- ONT SDD Document

6. APPENDIX – Database Test Results

Sign-up backend test: (type:1 admin , 2 sub-admin, 3 member) all of them gives OK status.



http://localhost:8080/api/addmember

POST http://localhost:8080/api/addmember

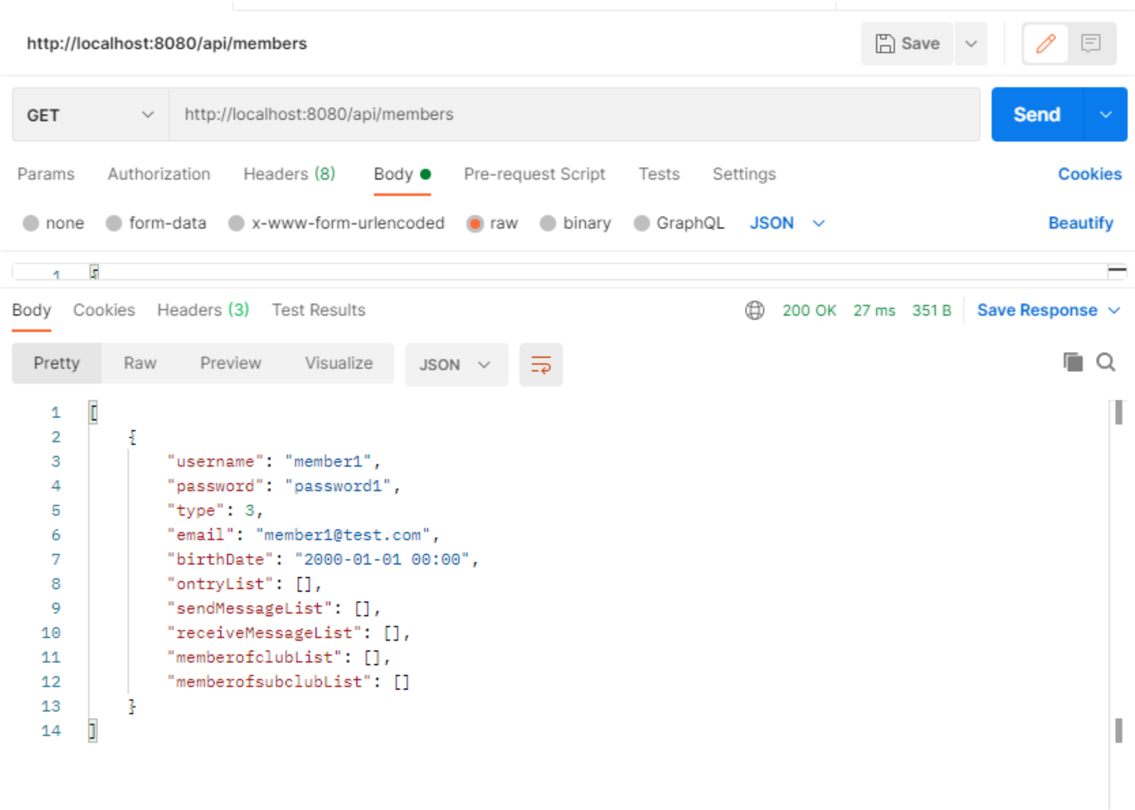
Body: raw JSON

```
{
    "username":"member1",
    "password":"password1",
    "type":3,
    "email":"member1@test.com",
    "birthDate":"2000-01-01 00:00"
}
```

200 OK 22 ms 145 B

```
{
    "status": "OK"
}
```

GET function test after Sign-up: (/api/admins for admins and /api/subadmins for sub-admins gives status OK)



http://localhost:8080/api/members

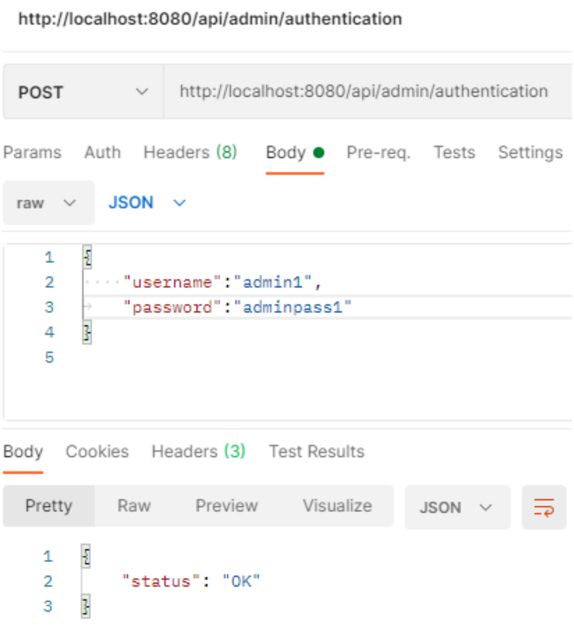
GET http://localhost:8080/api/members

Body: raw JSON

200 OK 27 ms 351 B

```
[
    {
        "username": "member1",
        "password": "password1",
        "type": 3,
        "email": "member1@test.com",
        "birthDate": "2000-01-01 00:00",
        "ontryList": [],
        "sendMessageList": [],
        "receiveMessageList": [],
        "memberofclubList": [],
        "memberofsubclubList": []
    }
]
```

Log-in (Authentication) test: Screenshots for admin but works for other types of users as well.

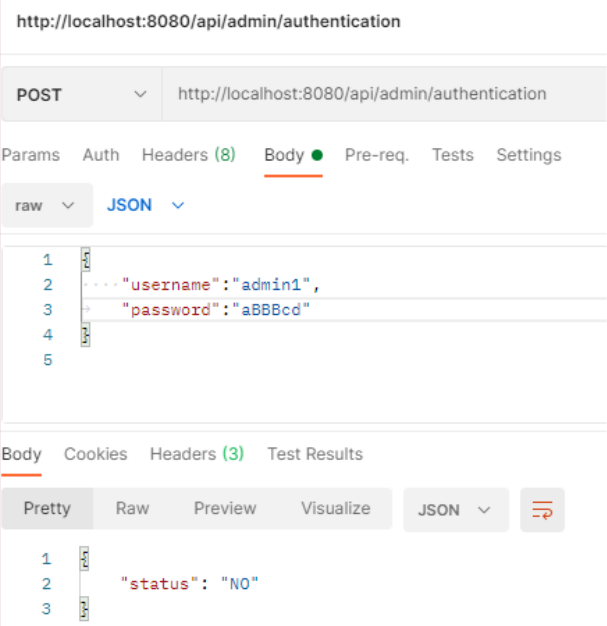


http://localhost:8080/api/admin/authentication

POST http://localhost:8080/api/admin/authentication

```
{
····"username":"admin1",
    "password":"adminpass1"
}
```

```
{
    "status": "OK"
}
```



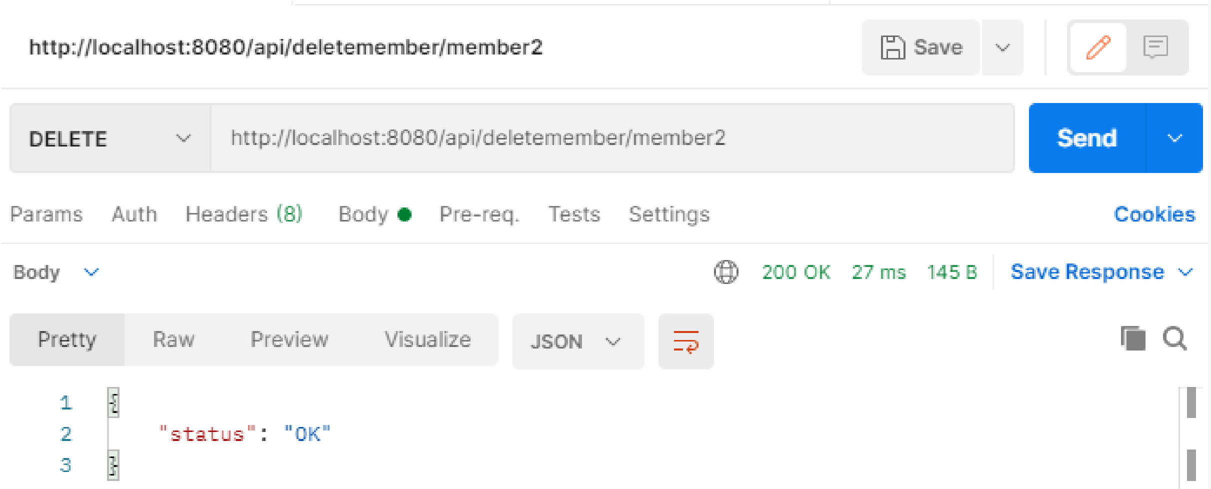
http://localhost:8080/api/admin/authentication

POST http://localhost:8080/api/admin/authentication

```
{
····"username":"admin1",
    "password":"aBBBcd"
}
```

```
{
    "status": "NO"
}
```

Delete user test:

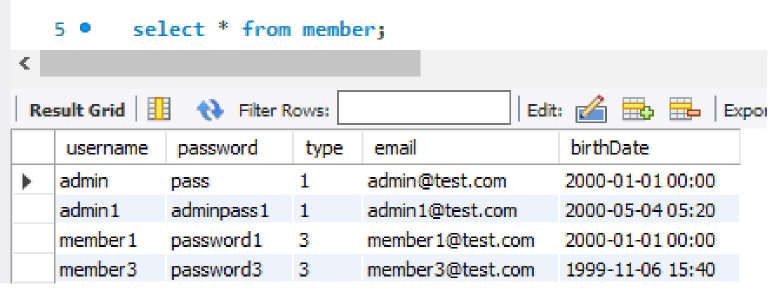


http://localhost:8080/api/deletemember/member2

DELETE http://localhost:8080/api/deletemember/member2

200 OK 27 ms 145 B

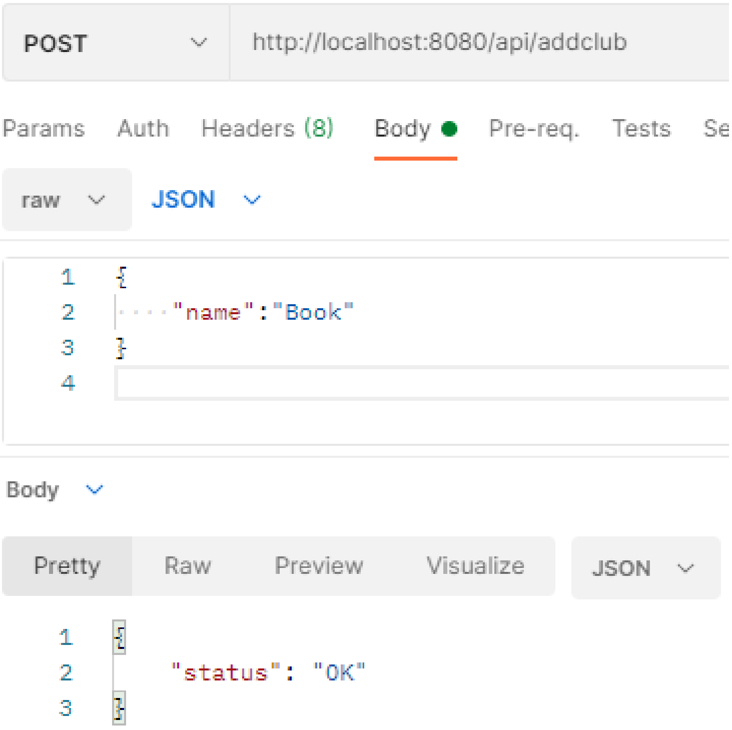
```
{
    "status": "OK"
}
```



```
5 ●    select * from member;
```

	username	password	type	email	birthDate
▶	admin	pass	1	admin@test.com	2000-01-01 00:00
	admin1	adminpass1	1	admin1@test.com	2000-05-04 05:20
	member1	password1	3	member1@test.com	2000-01-01 00:00
	member3	password3	3	member3@test.com	1999-11-06 15:40

Add Club test:

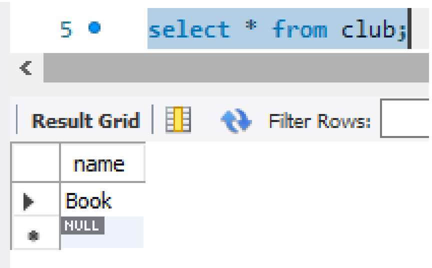


POST http://localhost:8080/api/addclub

JSON

```
{
····"name":"Book"
}
```

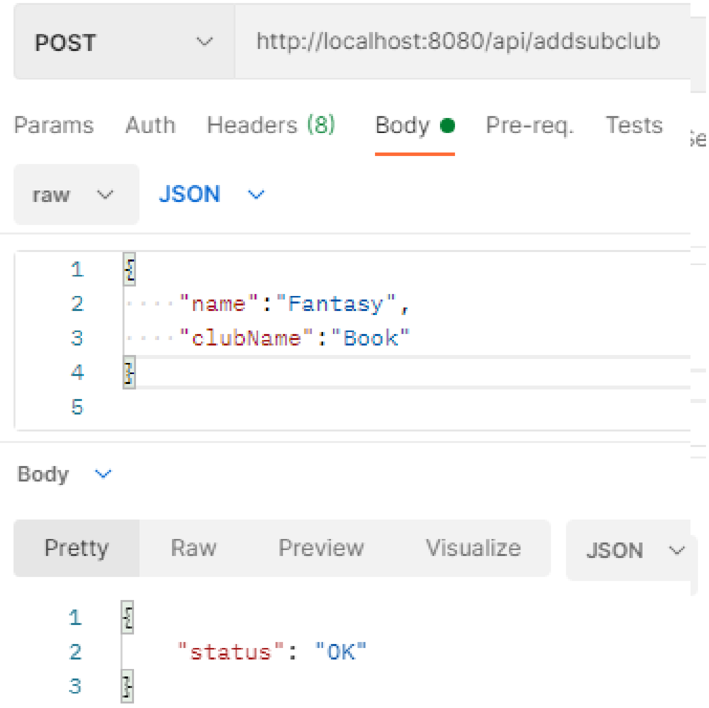
```
{
    "status": "OK"
}
```



```
5 ●  select * from club;
```

name
Book
NULL

Add Sub-club test:

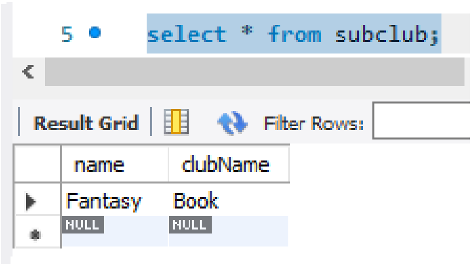


POST http://localhost:8080/api/addsubclub

JSON

```
{
····"name":"Fantasy",
····"clubName":"Book"
}
```

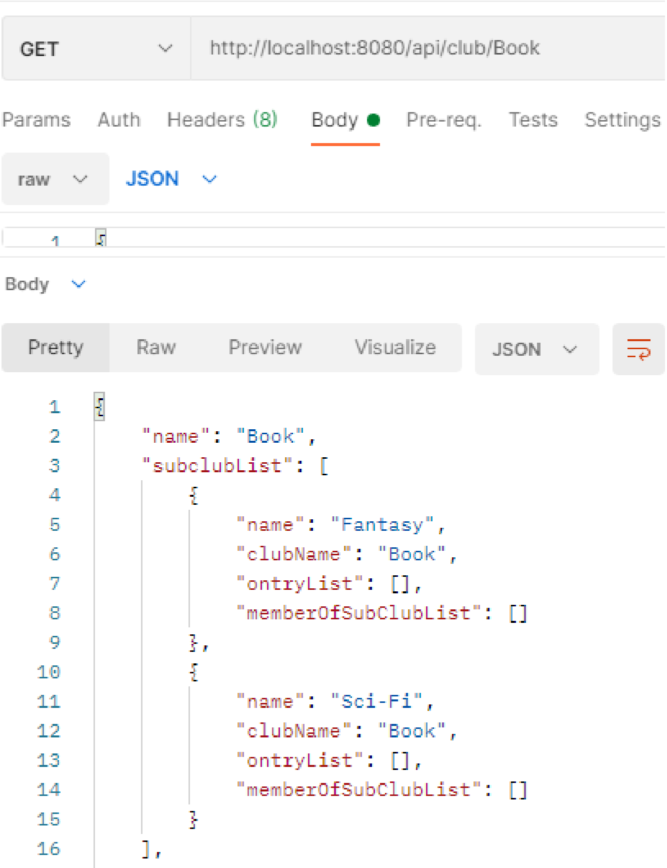
```
{
    "status": "OK"
}
```



```
5 ●  select * from subclub;
```

name	clubName
Fantasy	Book
NULL	NULL

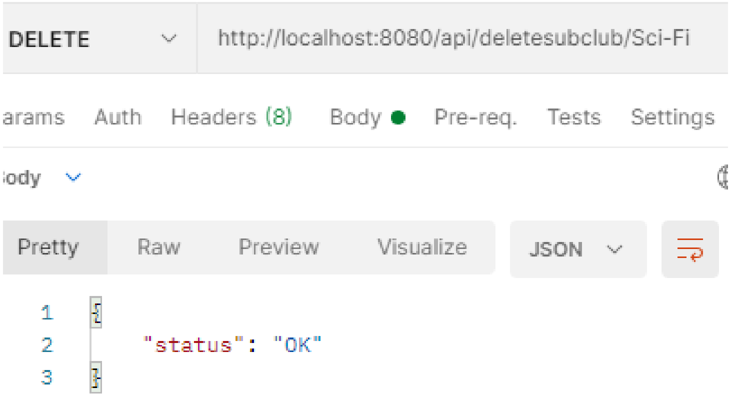
Club and Sub-club Relationship test:



GET http://localhost:8080/api/club/Book

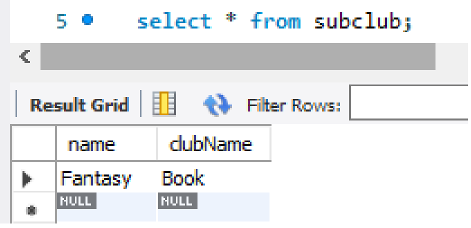
```
{
    "name": "Book",
    "subclubList": [
        {
            "name": "Fantasy",
            "clubName": "Book",
            "ontryList": [],
            "memberOfSubClubList": []
        },
        {
            "name": "Sci-Fi",
            "clubName": "Book",
            "ontryList": [],
            "memberOfSubClubList": []
        }
    ],
```

Delete Sub-club test:



DELETE http://localhost:8080/api/deletesubclub/Sci-Fi

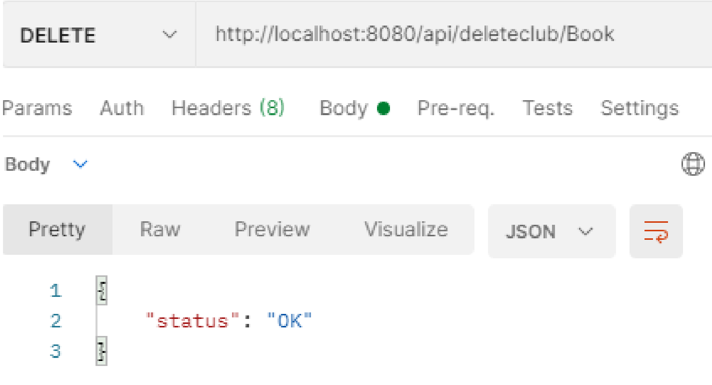
```
{
    "status": "OK"
}
```



```
select * from subclub;
```

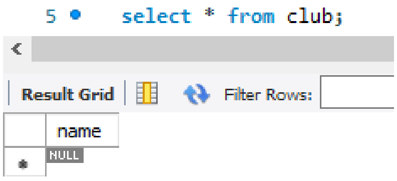
name	clubName
Fantasy	Book
NULL	NULL

Delete Club test:



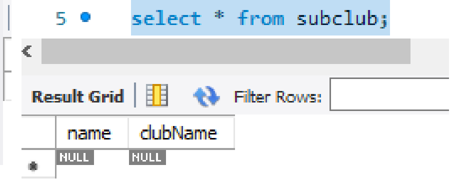
DELETE http://localhost:8080/api/deleteclub/Book

```
{
    "status": "OK"
}
```



```
5 ● select * from club;
```

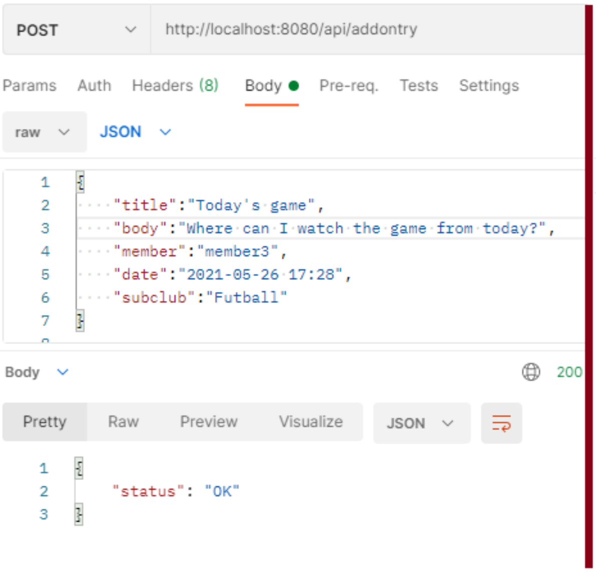
name
NULL



```
5 ● select * from subclub;
```

name	clubName
NULL	NULL

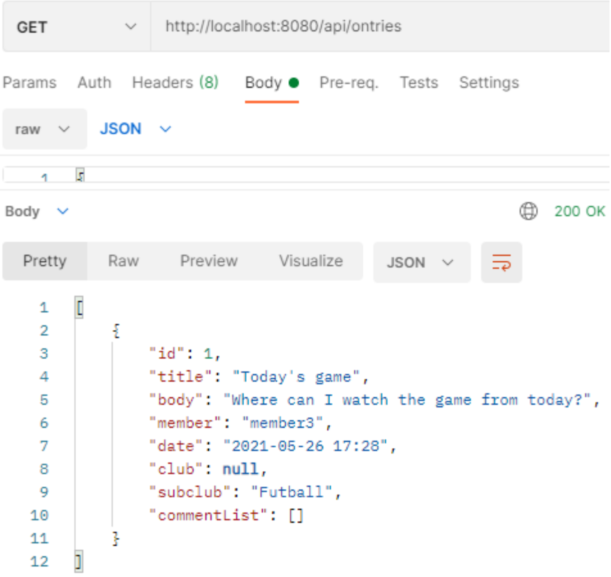
Post ontry to a sub-club test:



POST http://localhost:8080/api/addontry

```
{
    "title":"Today's game",
    "body":"Where can I watch the game from today?",
    "member":"member3",
    "date":"2021-05-26 17:28",
    "subclub":"Futball"
}
```

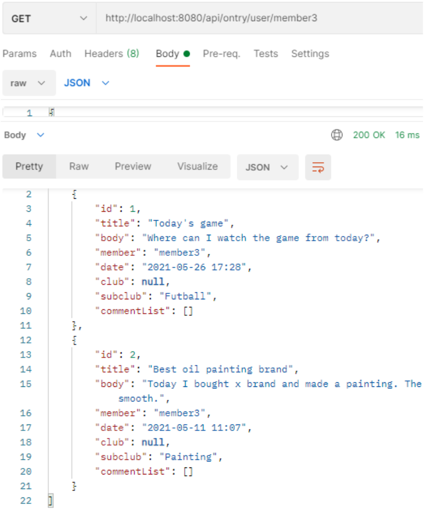
```
{
    "status": "OK"
}
```



GET http://localhost:8080/api/ontries

```
[
    {
        "id": 1,
        "title": "Today's game",
        "body": "Where can I watch the game from today?",
        "member": "member3",
        "date": "2021-05-26 17:28",
        "club": null,
        "subclub": "Futball",
        "commentList": []
    }
]
```

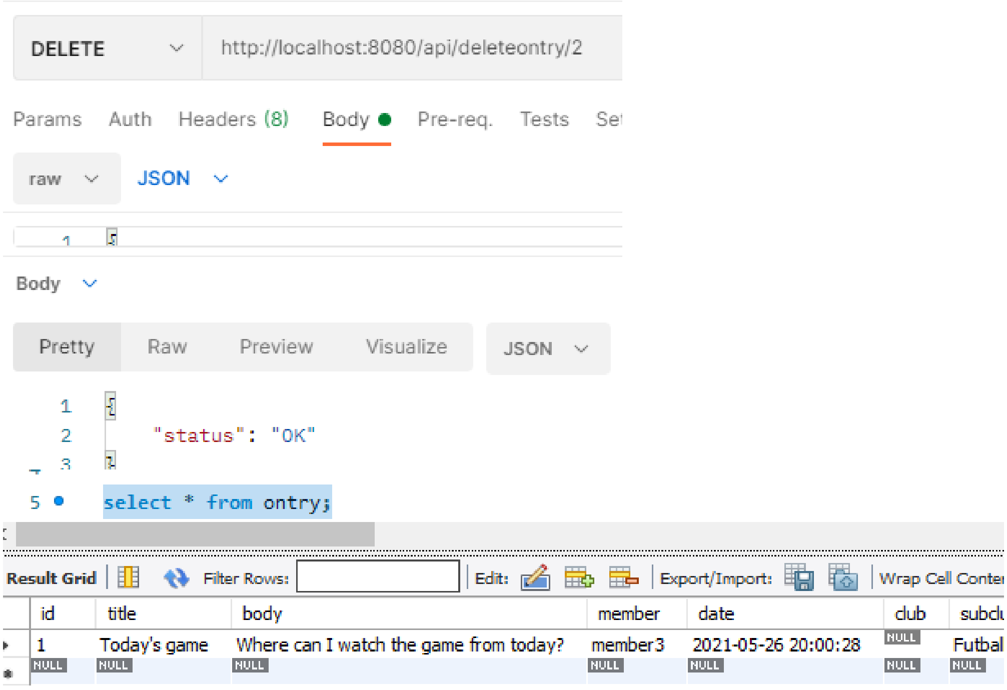
See ontries of a member test:



GET http://localhost:8080/api/ontry/user/member3

```
    {
        "id": 1,
        "title": "Today's game",
        "body": "Where can I watch the game from today?",
        "member": "member3",
        "date": "2021-05-26 17:28",
        "club": null,
        "subclub": "Futball",
        "commentList": []
    },
    {
        "id": 2,
        "title": "Best oil painting brand",
        "body": "Today I bought x brand and made a painting. The
            smooth.",
        "member": "member3",
        "date": "2021-05-11 11:07",
        "club": null,
        "subclub": "Painting",
        "commentList": []
    }
]
```

Delete ontry test:



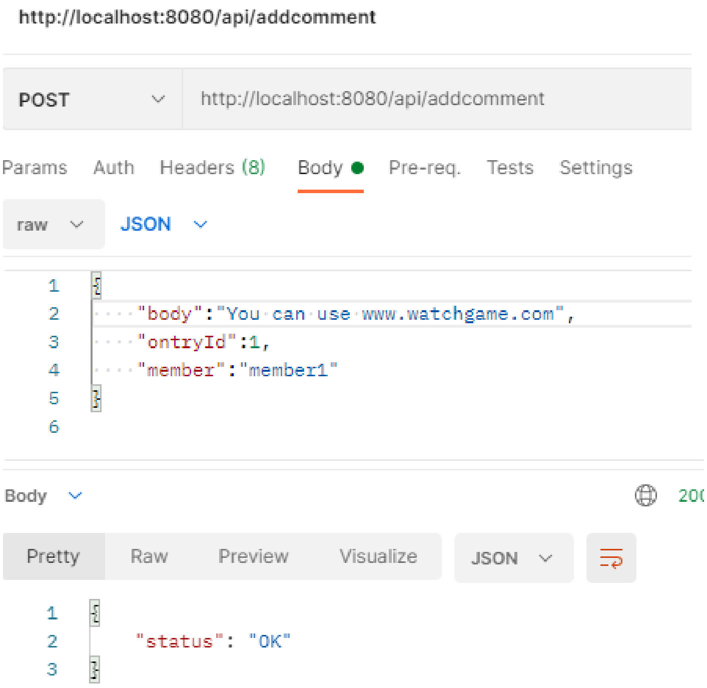
```
{
```

```
{
    "status": "OK"
}
```

```
select * from ontry;
```

id	title	body	member	date	club	subcl
1	Today's game	Where can I watch the game from today?	member3	2021-05-26 20:00:28	NULL	Futbal
NULL	NULL	NULL	NULL	NULL	NULL	NULL

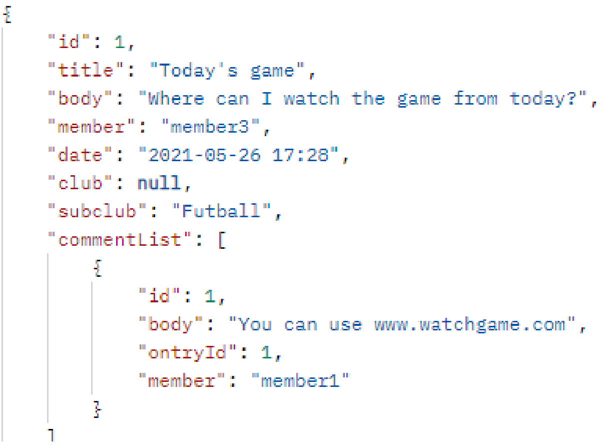
Add comment on a post test:



http://localhost:8080/api/addcomment
POST
http://localhost:8080/api/addcomment
Params
Auth
Headers (8)
Body
Pre-req.
Tests
Settings
raw
JSON

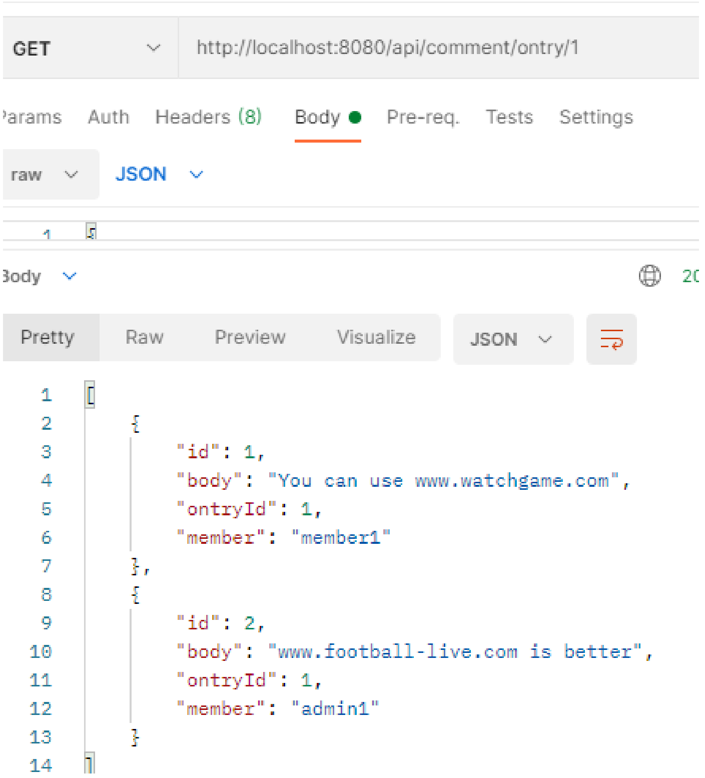
```
{
    "body":"You can use www.watchgame.com",
    "ontryId":1,
    "member":"member1"
}
```

```
{
    "status": "OK"
}
```



```
{
    "id": 1,
    "title": "Today's game",
    "body": "Where can I watch the game from today?",
    "member": "member3",
    "date": "2021-05-26 17:28",
    "club": null,
    "subclub": "Futball",
    "commentList": [
        {
            "id": 1,
            "body": "You can use www.watchgame.com",
            "ontryId": 1,
            "member": "member1"
        }
    ]
```

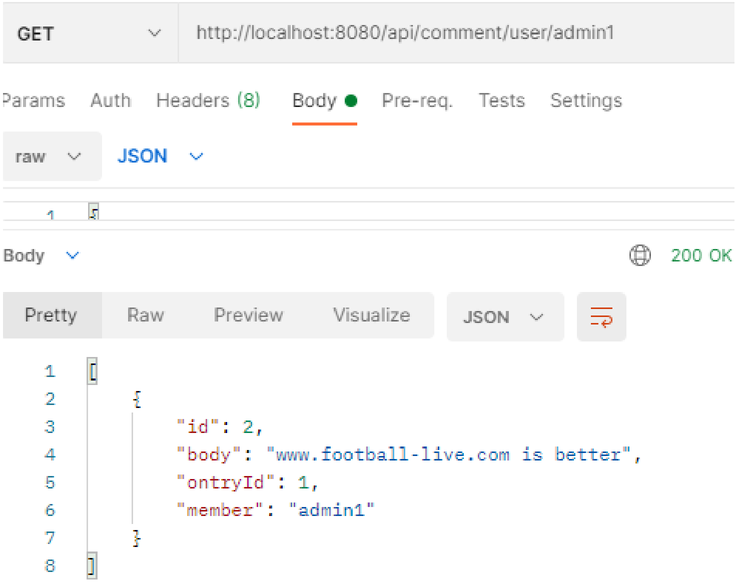
Comments on a post test:



GET http://localhost:8080/api/comment/ontry/1

```
[
    {
        "id": 1,
        "body": "You can use www.watchgame.com",
        "ontryId": 1,
        "member": "member1"
    },
    {
        "id": 2,
        "body": "www.football-live.com is better",
        "ontryId": 1,
        "member": "admin1"
    }
]
```

To see a user’s comments test:

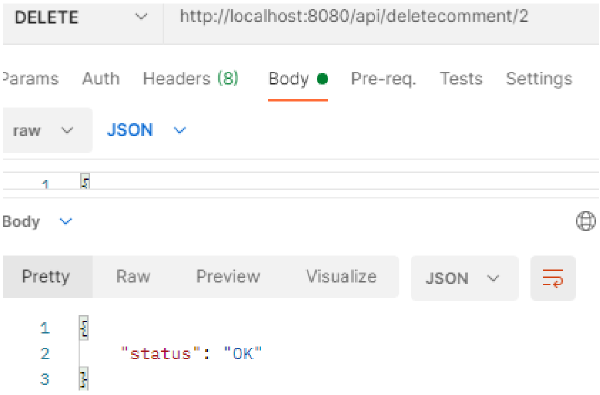


GET http://localhost:8080/api/comment/user/admin1

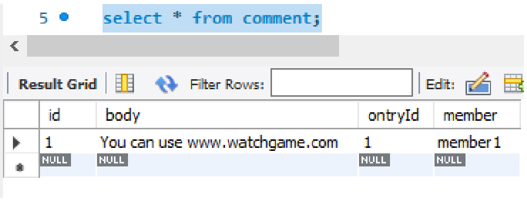
200 OK

```
[
    {
        "id": 2,
        "body": "www.football-live.com is better",
        "ontryId": 1,
        "member": "admin1"
    }
]
```

Delete comment test:

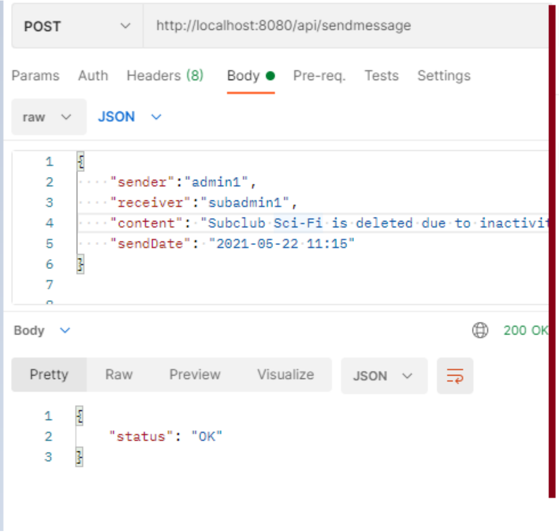

DELETE http://localhost:8080/api/deletecomment/2

```
{
    "status": "OK"
}
```


```
select * from comment;
```

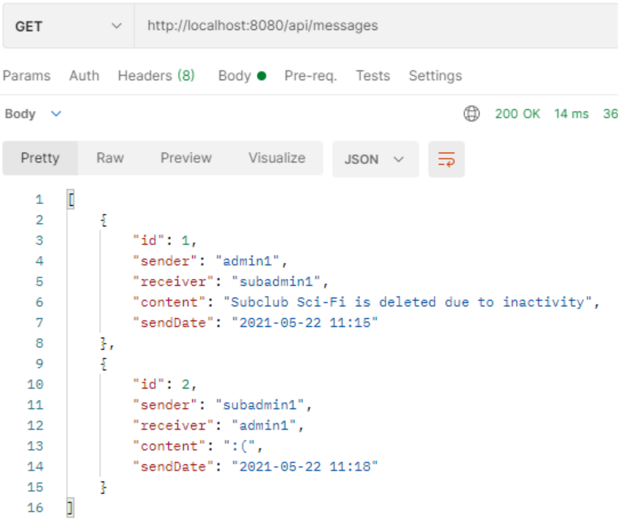
id	body	ontryId	member
1	You can use www.watchgame.com	1	member1
NULL	NULL	NULL	NULL

Send message test:


POST http://localhost:8080/api/sendmessage

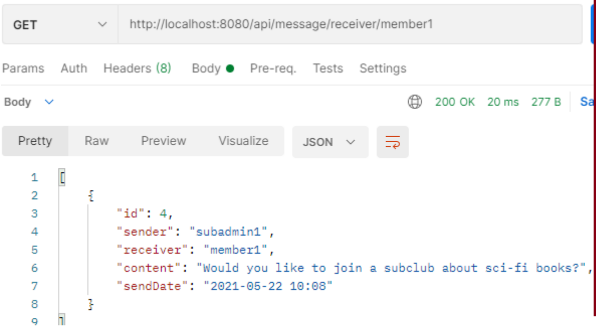
```
{
    "sender":"admin1",
    "receiver":"subadmin1",
    "content": "Subclub Sci-Fi is deleted due to inactivi
    "sendDate": "2021-05-22 11:15"
}
```

```
{
    "status": "OK"
}
```

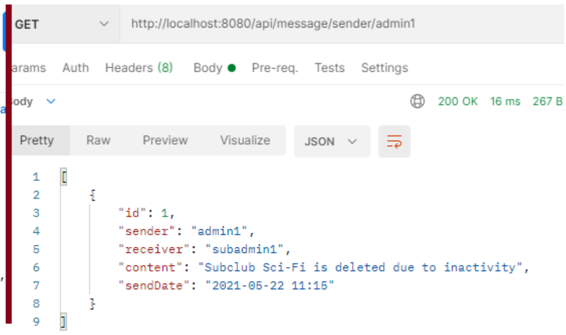

GET http://localhost:8080/api/messages

```
[
    {
        "id": 1,
        "sender": "admin1",
        "receiver": "subadmin1",
        "content": "Subclub Sci-Fi is deleted due to inactivity",
        "sendDate": "2021-05-22 11:15"
    },
    {
        "id": 2,
        "sender": "subadmin1",
        "receiver": "admin1",
        "content": ":(",
        "sendDate": "2021-05-22 11:18"
    }
]
```

See messages of a user(inbox) and messages from a user(sent) test:

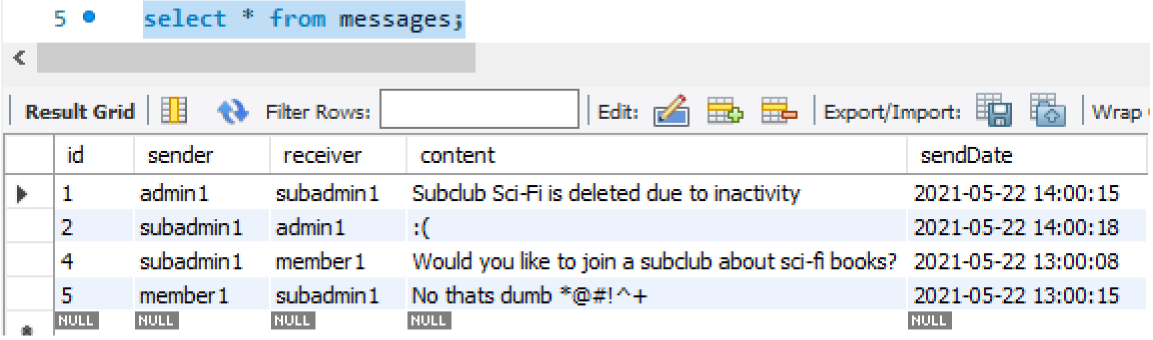

GET http://localhost:8080/api/message/receiver/member1

```
[
    {
        "id": 4,
        "sender": "subadmin1",
        "receiver": "member1",
        "content": "Would you like to join a subclub about sci-fi books?",
        "sendDate": "2021-05-22 10:08"
    }
]
```


GET http://localhost:8080/api/message/sender/admin1

```
[
    {
        "id": 1,
        "sender": "admin1",
        "receiver": "subadmin1",
        "content": "Subclub Sci-Fi is deleted due to inactivity",
        "sendDate": "2021-05-22 11:15"
    }
]
```

Delete messages test:

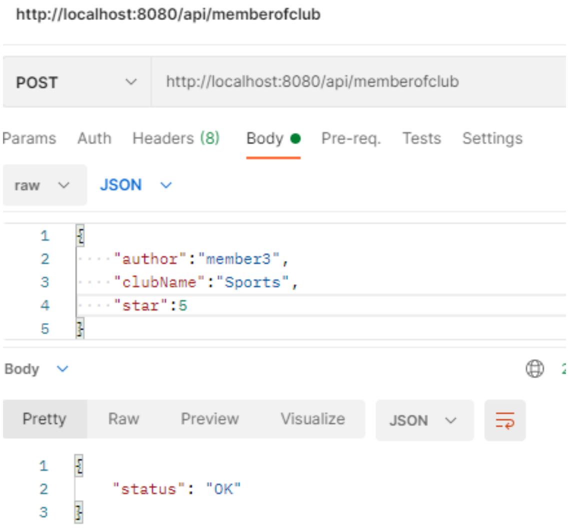


```
select * from messages;
```

id	sender	receiver	content	sendDate
1	admin1	subadmin1	Subclub Sci-Fi is deleted due to inactivity	2021-05-22 14:00:15
2	subadmin1	admin1	:(	2021-05-22 14:00:18
4	subadmin1	member1	Would you like to join a subclub about sci-fi books?	2021-05-22 13:00:08
5	member1	subadmin1	No thats dumb *@#!^+	2021-05-22 13:00:15
NULL	NULL	NULL	NULL	NULL

As you can see no message with ID:3 so its deleted.

Enrolling to a club and rating club:



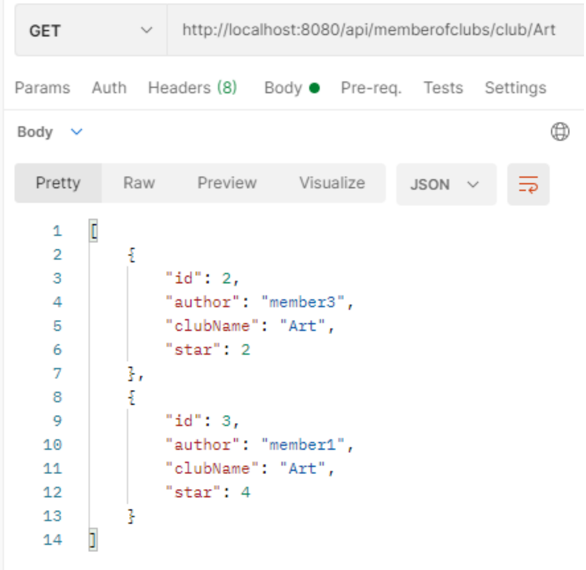
http://localhost:8080/api/memberofclub

POST http://localhost:8080/api/memberofclub

```
{
    "author":"member3",
    "clubName":"Sports",
    "star":5
}
```

```
{
    "status": "OK"
}
```

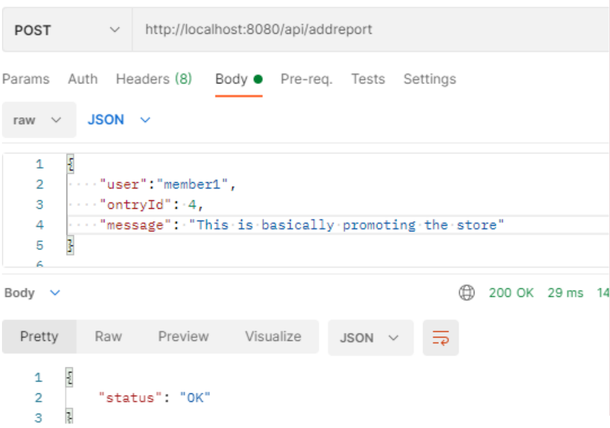
Club member list test:



GET http://localhost:8080/api/memberofclubs/club/Art

```
[
    {
        "id": 2,
        "author": "member3",
        "clubName": "Art",
        "star": 2
    },
    {
        "id": 3,
        "author": "member1",
        "clubName": "Art",
        "star": 4
    }
]
```

Report ontry test:

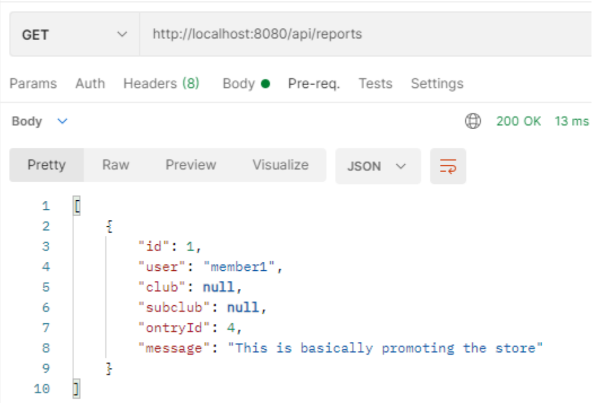


POST http://localhost:8080/api/addreport

```
{
    "user":"member1",
    "ontryId": 4,
    "message": "This is basically promoting the store"
}
```

200 OK 29 ms 14

```
{
    "status": "OK"
}
```

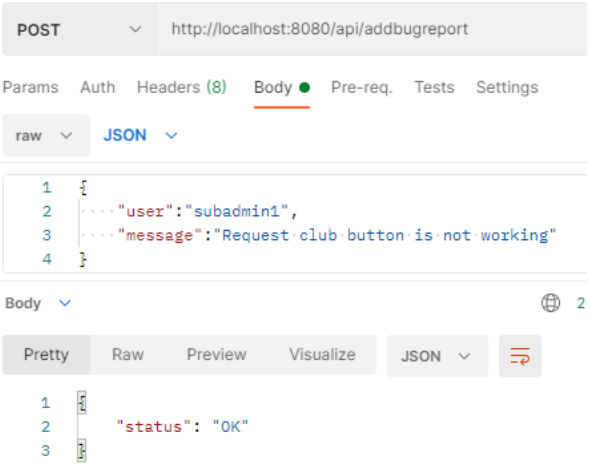


GET http://localhost:8080/api/reports

200 OK 13 ms

```
[
    {
        "id": 1,
        "user": "member1",
        "club": null,
        "subclub": null,
        "ontryId": 4,
        "message": "This is basically promoting the store"
    }
]
```

Bug report test:



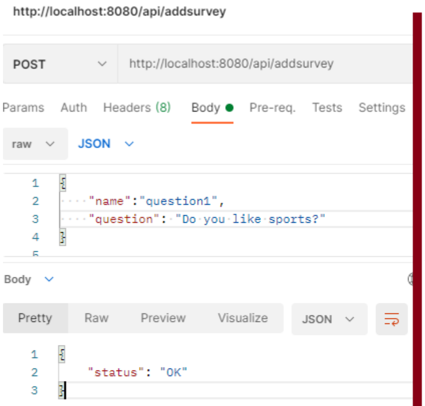
POST http://localhost:8080/api/addbugreport

```
{
    "user":"subadmin1",
    "message":"Request club button is not working"
}
```

2

```
{
    "status": "OK"
}
```

Questionnaire test:

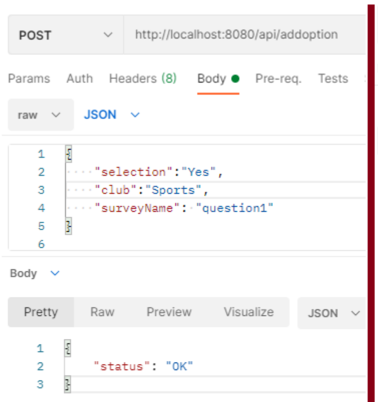


http://localhost:8080/api/addsurvey

POST http://localhost:8080/api/addsurvey

```
{
    "name":"question1",
    "question": "Do you like sports?"
}
```

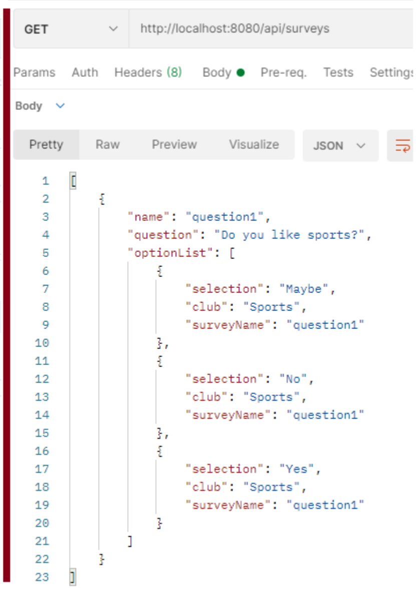
```
{
    "status": "OK"
}
```



POST http://localhost:8080/api/addoption

```
{
    "selection":"Yes",
    "club":"Sports",
    "surveyName": "question1"
}
```

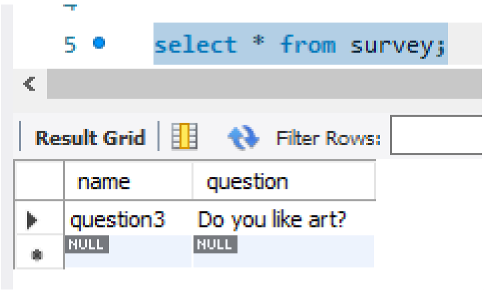
```
{
    "status": "OK"
}
```



GET http://localhost:8080/api/surveys

```
[
    {
        "name": "question1",
        "question": "Do you like sports?",
        "optionList": [
            {
                "selection": "Maybe",
                "club": "Sports",
                "surveyName": "question1"
            },
            {
                "selection": "No",
                "club": "Sports",
                "surveyName": "question1"
            },
            {
                "selection": "Yes",
                "club": "Sports",
                "surveyName": "question1"
            }
        ]
    }
]
```

Delete question from questionnaire test:



```
select * from survey;
```

name	question
question3	Do you like art?
NULL	NULL

As you can see question 1 and 2 are deleted.